

EXHIBIT B

The AdviceNet System

Sunday, January 26, 1997

The Subscription Editor

I'm thinking we should store each advice subscription as a separate document, which would contain at least these parts:

- ☐ the site's URL
- ☐ the site's title
- ☐ a short description of the site
- ☐ the site's recommended update frequency
- ☐ the user's chosen update frequency

We'll need a small application which displays this information, allows the frequency to be edited, and can trigger an immediate update. It should probably have an "Update All" switch as well.

This would also be the application to use to create a new subscription by hand, and it should be tied in to Internet Config so that it can display advice sites when other programs find them on the net.

Perhaps this application should be able to make some changes to the advice database as well, such as purging advice from a particular site or adding new advice from files on the local file system.

The Subscription Demon

This demon (which may be in the same background application as the information demon) watches for the subscription list to change or for subscriptions to get out of date. When they do, it tells the advice gatherer to update the advice from the subscription's URL.

The Advice Gatherer

This part goes to the network in response to requests from the subscription editor ("Update Now"), the subscription demon (as subscriptions go out of date), or the information demon (as pieces of advice recommend gathering others). It downloads pieces of advice from the net, parses them, and links their guards into the information network.

Ideally the gatherer would have the usual array of internet protocols at its disposal, but I don't see any reason why HTTP, SHTTP, and local file system access wouldn't suffice.

The Fact Checkers

These parts are called upon by advice guards to examine the state of the system. Each will know how to extract some piece of information from the system, and will have mechanisms for reviewing that information. We should have a plug-in system for these pieces, so that people can check for information we haven't yet imagined. But we should also include some standard kinds of information:

- ☐ devices
- ☐ computer & CPU model
- ☐ amount of RAM
- ☐ directory and file info
- ☐ system components: applications, control panels, fonts, etc.
- ☐ version numbers
- ☐ gestalt values
- ☐ advice subscriptions
- ☐ advice stored in the system
- ☐ advice which has been triggered

The Information Network

The information network builds complex facts out of the simple facts and relationships described by the fact checkers. Each node of the network will represent some fact about the system, and links between the nodes will carry updates to the information. When the advice gatherer collects a piece of advice, the guard on that advice will be linked into the network, along with any lesser conditions it depends upon. To keep the network efficient, different guards will be able to share nodes in the network — there may be hundreds of pieces of advice which refer to the system version number, but the version number will have only one node in the network.

The Information Demon

This demon scans the information network, watching for conditions which have changed and updating the facts related to them. When it finds that a guard condition is satisfied, it may ask the gatherer for more advice, or launch the display application to display advice to the user. (Before displaying advice, it should probably flush the information network for more advice that has become pertinent.)

The Display Application

The display application is launched by the information demon when a piece of advice becomes pertinent. It will display the advice's message (presumably in HTML, but we ought to support some other MIME types) and the associated signature.

It will also provide an extension — presumably a new form submission method — which can trigger effects not usually available to HTML. We want advice, once accepted, to be able to download and execute an arbitrary native application.

The HTML will usually contain the URLs and checksum of an application to run, and of files to be handed to that application as input. In addition, the application will receive the form result, if any. Through this mechanism and the use of hidden fields the effect application can also get information on how the advice was triggered.

In addition to the options the advice offers to the user, the display application should provide some standard options. When the user sees a piece of advice, he should always be able to postpone it, ignore it, have the system forget it, check for updates to it, or cancel his subscription to the site it came from.

Finally, the display application should be able to display and check the digital signatures on advice.

The Effects

Once accepted, advice acts on the system by launching applications. Any application could be launched as the effect of a piece of advice, but I imagine that some small applications will be written specifically for advice, and thus take advantage of information about the advice which we make available to them. These applications can be written in either a traditional programming language or in any general-purpose scripting language available on the target system.

While a piece of advice can download its effect application from the net, it's best if the applications for executing some common kinds of advice are available locally. The only three kinds of applications I've thought of that we need are downloading files to install, locating misplaced files, and changing advice subscriptions.

Central Advice Databases

There are a few kinds of advice we should probably provide ourselves, either as part of the bootstrapping process or as a service to enhance the usability of AdviceNet:

- Site Announcements, which would contain advice like "If you have MacWrite, you may want to subscribe to the Claris advice site."
- Urgent Updates, containing advice like "If you subscribe to the Claris site, and you downloaded advice from it yesterday, you should check it again immediately, because they told us they published something really stupid."
- The Better Advice Bureau, which gives advice like "If you subscribe to EvilCorp, you should know that we get lots of complaints about the advice they give out. We think you should stop your subscription and purge their advice from your system."

Automated Advice Writers

Some kinds of advice can be so detailed that it's much easier to extract the right advice from a working machine than it is to write it by hand. In particular, a thorough check of a complicated software installation may be difficult to write by hand. Also, some kinds of advice, such as a check of the system and hardware requirements for a piece of software, could be written just by filling out a form. So we should write a few applications which produce advice by examining part of an example machine, or by questioning the advice author. The output can then be tweaked into the advice the user needs.

Direct Advice Installation

We'd like to provide a mechanism for a software installer to directly add advice to the database. That way the big lump of advice that comes with a new software package won't require a lengthy download.